TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN
TRABAJO PRÁCTICO INTEGRADOR ESTADÍSTICA-PROGRAMACIÓN II

Integrantes:

Augusto Camani

Santiago Feresin

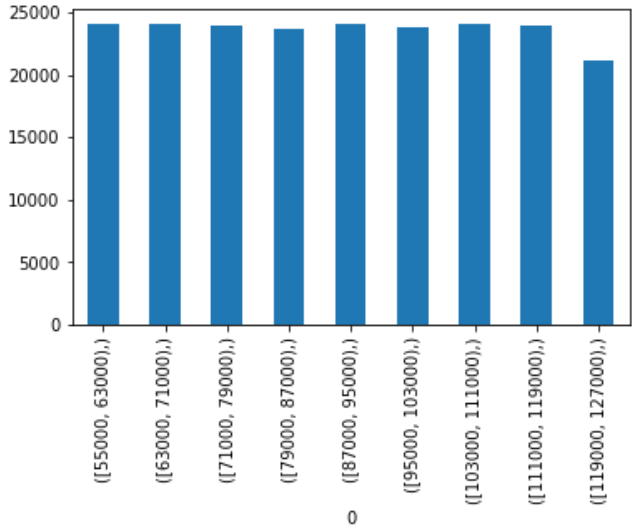
Nicolás Martinez

Practica:

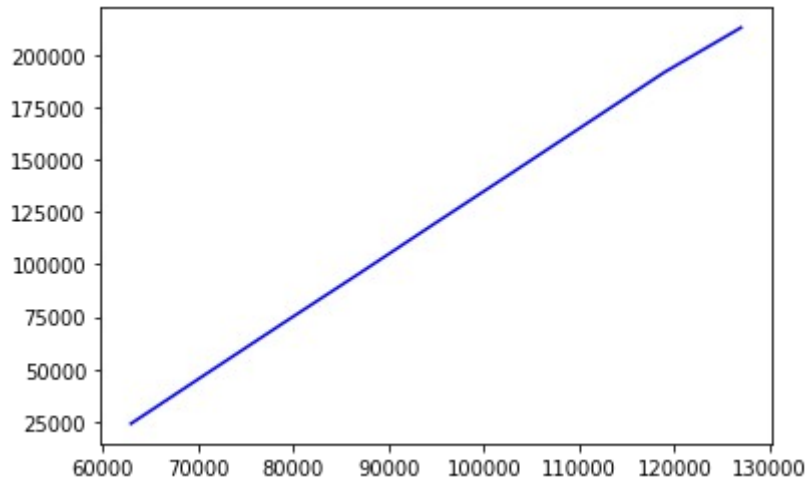
a) Construir una tabla de frecuencias.

	x	f	F	fr	Fr
0	55000	24087	24087	0.11309	0.11
1	63000	24070	48157	0.11301	0.23
2	71000	23990	72147	0.11263	0.34
3	79000	23721	95868	0.11137	0.45
4	87000	24088	119956	0.11309	0.56
5	95000	23867	143823	0.11206	0.68
6	103000	24003	167826	0.11269	0.79
7	111000	23980	191806	0.11259	0.90
8	119000	21186	212992	0.09947	1.00

b) Construir el histograma de frecuencias relativas absoluta y acumulada.



c) Construir el polígono de frecuencias relativas acumuladas.



d) Calcular las medidas de tendencia central e interpretarlas.

Media: 90508.47110220102

Mediana: 90547.0

Moda: [58550, 71331, 113294, 115473]

e) Obtener las medidas de dispersión e interpretarlas.

Varianza: 421244853.7043449

Desvío estándar: 20524.250381057645

Cuartil 1: 72692.75

Cuartil 3: 108306.0

Rango intercuartilico: 35613.25

Coeficiente de variación: 22.676607096679298

f) Calcular el P30. ¿Qué significa este resultado?

Percentil 30: 76254.30

El percentil sub 30 significa que debajo de ese número se divide la muestra en porcentajes: por debajo del número se encuentra el 30% de los datos de la muestra y por encima el 70% de la muestra.

g) ¿Qué porcentaje de automóviles realizan la VTV con menos de 70.000 km?

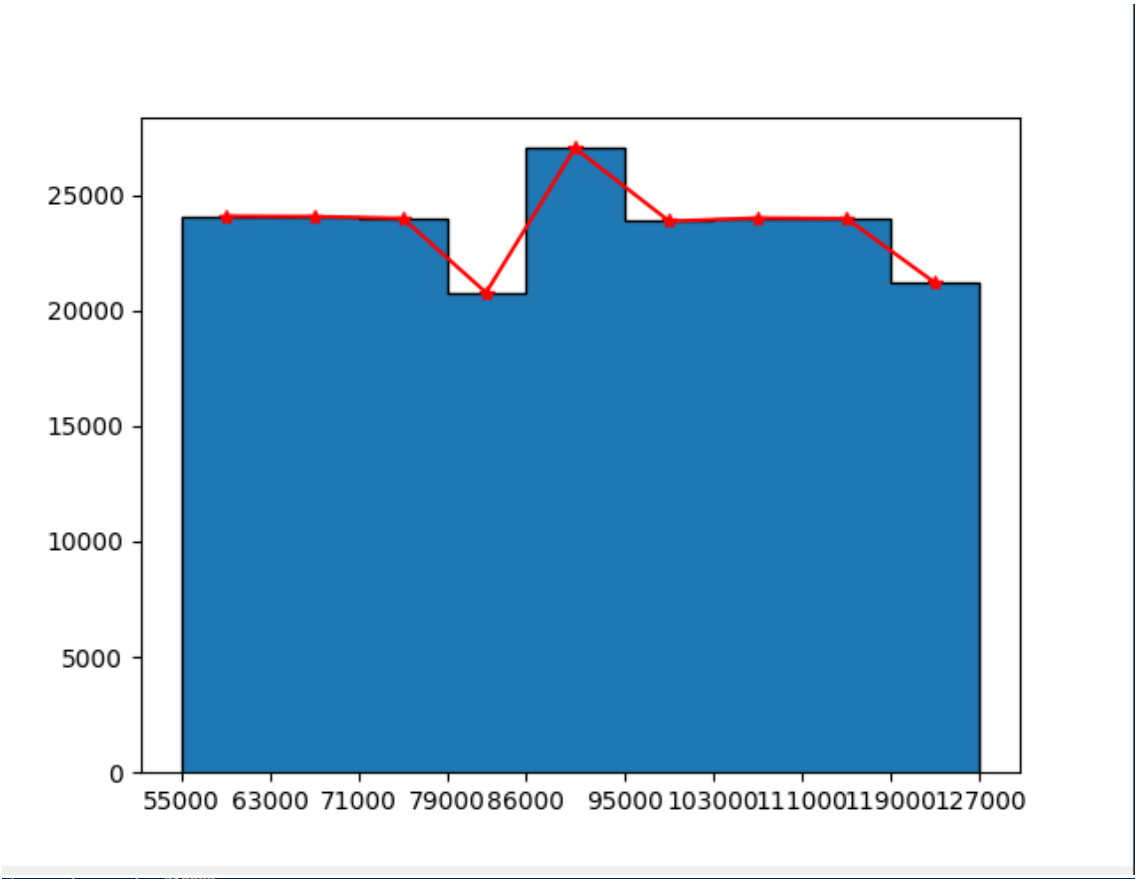
Los autos que realizan la VTV con menos de 70 mil km son: 45179

El porcentaje de autos que realizan la VTV con menos de 70 mil km es: 21.21%

h) Analizar la simetría o asimetría de la distribución de frecuencias resultante.

El coeficiente de asimetría de Pearson es de: 0.00027. Podemos afirmar que la muestra es casi simétrica, y su distribución es pareja ya que el coeficiente de asimetría es un número ínfimo.

Códigos:




```
La muestra es de: 212992
La sumatoria de frecuencias es: 212992
La cantidad de elementos no repetidos son: 67513
El valor mínimo es: 55000
El valor máximo es: 126000
Datos agrupados en intervalos de 8000 km.
Los autos que realizan la VTV con menos de 70 mil km son: 45179
El porcentaje de los autos que realizan la VTV con menos de 70 mil km son: 21.21159 %
Frecuencias por cada intervalo: [24087, 24070, 23990, 23721, 24088, 23867, 24003, 23980, 21186]
Frecuencias acumuladas por intervalos: [24087, 48157, 72147, 95868, 119956, 143823, 167826, 191806, 212992]
Frecuencias relativas por intervalos: [0.11309, 0.11301, 0.11263, 0.11137, 0.11309, 0.11206, 0.11269, 0.11259, 0.09947]
Frecuencias relativas acumuladas por intervalos: [0.11, 0.23, 0.34, 0.45, 0.56, 0.68, 0.79, 0.9, 1.0]
Esta es la moda más pequeña: 58550
Esta es la mediana: 90547.0
Esta es la media: 90508.47
Estas son todas las modas: [58550, 71331, 113294, 115473] C/U se repite 12 veces.
```

```
Esta es la varianza: 421244853.7043
Este es el desvío estandar: 20524.2504
Este es el coeficiente de variación: 22.68 %
Este es el coeficiente de asimetría de pearson: 0.0003
Posición de P sub 30 en la frecuencia absoluta acumulada. 63897.6
P sub 30, se encuentra en el tercer intervalo.
Por debajo de 76249 se encuentra el 30% de los datos.
```

	x	f	F	fr	Fr
0	55000	24087	24087	0.11309	0.11
1	63000	24070	48157	0.11301	0.23
2	71000	23990	72147	0.11263	0.34
3	79000	23721	95868	0.11137	0.45
4	87000	24088	119956	0.11309	0.56
5	95000	23867	143823	0.11206	0.68
6	103000	24003	167826	0.11269	0.79
7	111000	23980	191806	0.11259	0.90
8	119000	21186	212992	0.09947	1.00

```
df = pd.read_csv('data.csv', header=None)
res, bins = pd.cut(df[df.columns[0]],range(55000, 130000, 7100), right=False, retbins=True)
res = pd.DataFrame(res)
res
```

	0
0	[55000, 62100)
1	[62100, 69200)
2	[62100, 69200)
3	[104700, 111800)
4	[97600, 104700)
...	...
212987	[76300, 83400)
212988	[97600, 104700)
212989	[111800, 118900)
212990	[83400, 90500)
212991	[83400, 90500)

```
res.value_counts().to_csv('tabla_frecuencia.csv')
res.value_counts()
```

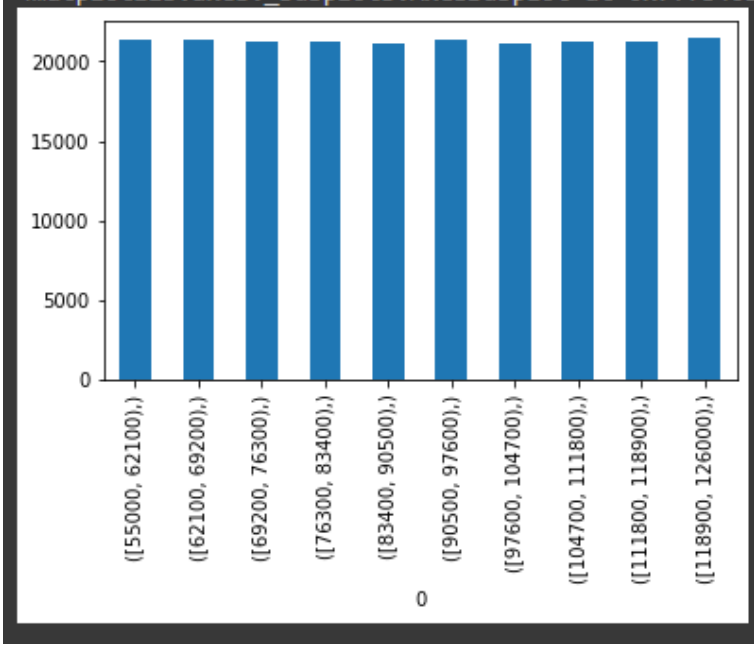
```
[118900, 126000)    21495
[90500, 97600)      21412
[55000, 62100)      21381
[62100, 69200)      21373
[104700, 111800)    21284
[69200, 76300)      21277
[111800, 118900)    21268
[76300, 83400)      21222
[97600, 104700)     21170
[83400, 90500)      21108
dtype: int64
```

```
samples = res.value_counts().sort_index()
samples.to_numpy().flatten()
```

```
array([21381, 21373, 21277, 21222, 21108, 21412, 21170, 21284, 21268,
       21495])
```

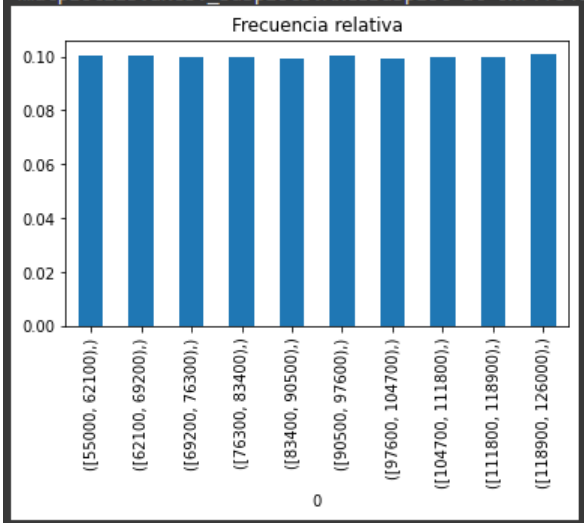
```
fig, ax = plt.subplots(figsize=(6, 3.5))
(
    pd.Series(samples).plot.bar(ax=ax)
)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff8403e7710>
```

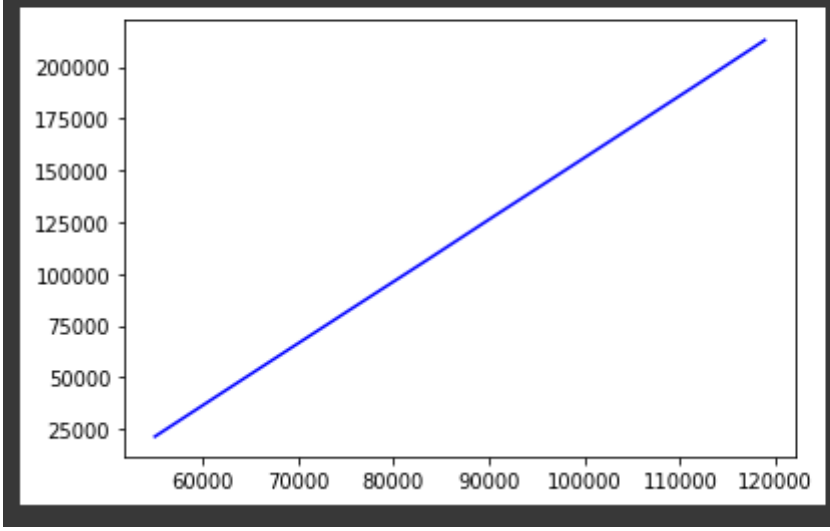



```
# Creamos histograma de frecuencias
fig, ax = plt.subplots(figsize=(6, 3.5))
(
    pd.Series(samples/samples.sum(axis=0)).plot.bar(ax=ax, title='Frecuencia relativa')
)
```

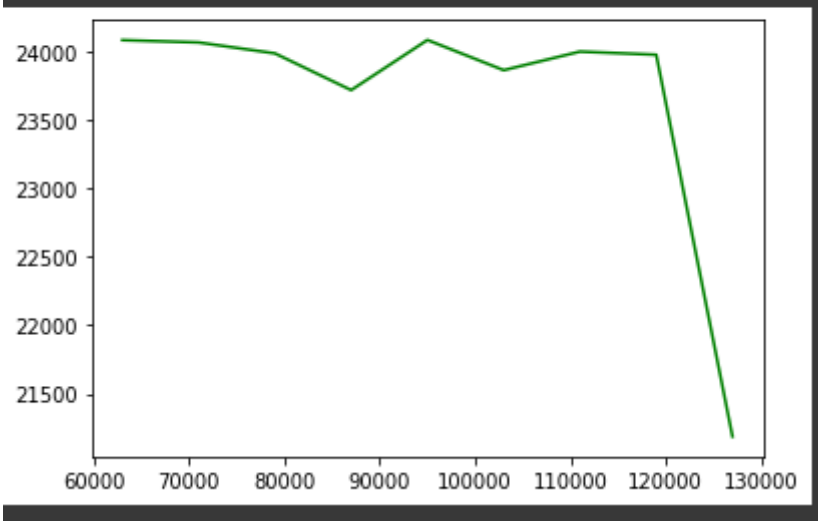
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff84029f110>
```



```
cumulative = np.cumsum(pd.Series(samples))
# Creamos grafico de frecuencias acumuladas
plt.plot(bins[:-1], cumulative, c='blue')
plt.show()
```




```
plt.plot(bins[1:], samples, c='green')
plt.show()
```



```
medidas_tendencia_central = {}
medidas_tendencia_central['Media'] = df[df.columns[0]].mean()
medidas_tendencia_central['Mediana'] = df[df.columns[0]].median()
medidas_tendencia_central['Moda'] = list(df[df.columns[0]].mode())
medidas_tendencia_central['Porcentaje de autos que realizan la VTV con menos de 70mil Km'] = 21.21159

medidas_tendencia_central
```

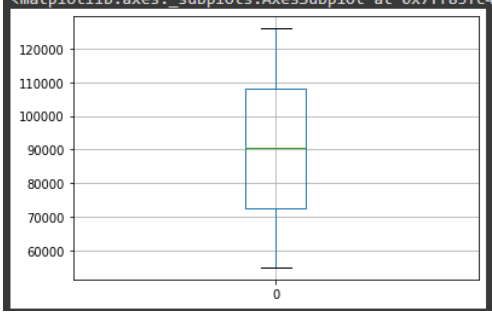
```
{'Media': 90508.47110220102,
 'Mediana': 90547.0,
 'Moda': [58550, 71331, 113294, 115473],
 'Porcentaje de autos que realizan la VTV con menos de 70mil Km': 21.21159}
```

```
medidas_dispersion = {}
medidas_dispersion['Varianza'] = df[df.columns[0]].var()
medidas_dispersion['Desvio estándar'] = df[df.columns[0]].std()
medidas_dispersion['Cuartil 1'] = df[df.columns[0]].quantile(0.25)
medidas_dispersion['Cuartil 3'] = df[df.columns[0]].quantile(0.75)
medidas_dispersion['Rango intercuartilico'] = df[df.columns[0]].quantile(0.75) - df[df.columns[0]].quantile(0.25)
medidas_dispersion['Coeficiente de variación'] = (df[df.columns[0]].std() / df[df.columns[0]].mean()) * 100
medidas_dispersion['Coeficiente de asimetría de Pearson'] = df[df.columns[0]].skew()
medidas_dispersion['P30'] = 69200 + ((((30 * 212992) / 100) - 42754) / 21277) * 7100

medidas_dispersion
```

```
{'Varianza': 421244853.7043449,
 'Desvio estándar': 20524.250381057645,
 'Cuartil 1': 72692.75,
 'Cuartil 3': 108306.0,
 'Rango intercuartilico': 35613.25,
 'Coeficiente de variación': 22.676607096679298,
 'Coeficiente de asimetría de Pearson': 0.0002707789859456798,
 'P30': 76255.48526577995}
```

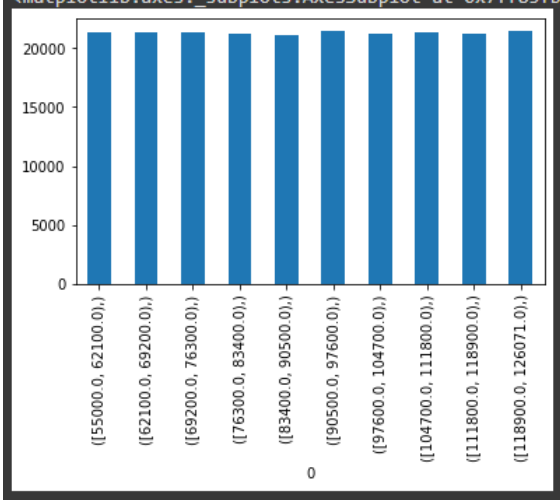
```
df.boxplot(column=[df.columns[0]])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff83fc494d0>
```



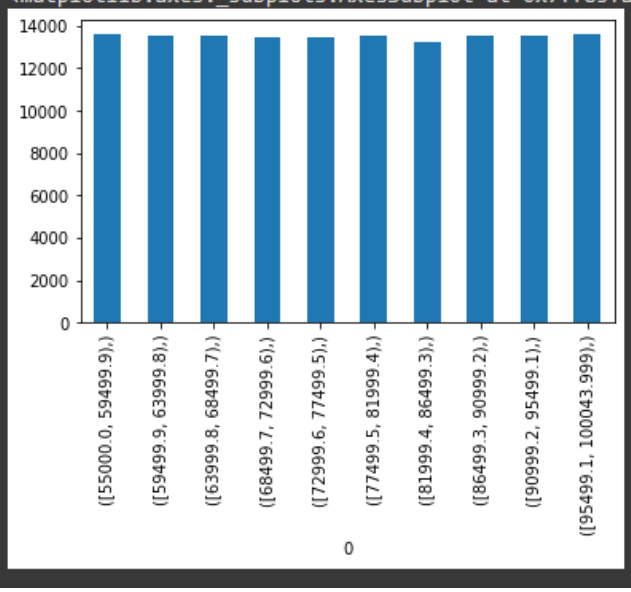
```python
#Quitamos valores extremos
df2 = df[df[df.columns[0]] > 10000]

res2, bins = pd.cut(df2[df2.columns[0]],10, right=False, retbins=True)
res2 = pd.DataFrame(res2)

samples2 = res2.value_counts().sort_index()
samples2.to_numpy().flatten()

fig, ax = plt.subplots(figsize=(6, 3.5))
(
    pd.Series(samples2.sort_index()).plot.bar(ax=ax)
)

#Ejercicio: Quitar los valores extremos grandes usando nombre variables df3, res3 y samples3
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff83fb5bfd0>
```



```
df3 = df[(df[df.columns[0]] < 100000) & (df[df.columns[0]] > 10000)]

res3, bins = pd.cut(df3[df3.columns[0]],10, right=False, retbins=True)
res3 = pd.DataFrame(res3)

samples3 = res3.value_counts().sort_index()
samples3.to_numpy().flatten()

fig, ax = plt.subplots(figsize=(6, 3.5))
(
    pd.Series(samples3.sort_index()).plot.bar(ax=ax)
)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff83fa59290>
```



Código de python hecho por nosotros:

```
import pandas as pd
from statistics import mean, median, mode, multimode , variance
import numpy as np
from scipy.stats import skew
import matplotlib.pyplot as plt

#Abro el archivo como una variable.
miArchivo = open("TUP2.txt", "r")

#Creo una variable como el archivo leído
datos = miArchivo.read()

#Por cada ";" que aparezca en el archvio, creo un elemento singular en la lista 'datosLista'.
listaDatos = datos.split(";")
listaDatos = list(map(int, listaDatos))

#Creamos la variable muestraTotal (Es la cantidad total de elementos con los que contamos).
muestraTotal = len(listaDatos)

#Ordenamos la datosLista(lista de todos los datos) de menor a mayor.
listaDatos.sort()

#Lista que contiene cada dato sin repetir.
listaValores = [listaDatos[0]]
```

```
#Guardamos el primer valor de la lista en un auxiliar.
auxiliar = 0

menoresA70 = 0

#Recorremos la lista y añadimos los elementos que no se repiten a una nueva lista.
for i in listaDatos:
    if i != auxiliar:
        listaValores.append(i)
        auxiliar = i
    if i < 70000:
        menoresA70 = menoresA70 + 1

porcentajeMenoresA70 = round((menoresA70 / muestraTotal) * 100 ,5)

#Variables para agrupados.
datosIntervalo1 = 0
datosIntervalo2 = 0
datosIntervalo3 = 0
datosIntervalo4 = 0
datosIntervalo5 = 0
datosIntervalo6 = 0
datosIntervalo7 = 0
datosIntervalo8 = 0
datosIntervalo9 = 0
frecuenciasAbsolutasIntervalos = []

for i in listaDatos:
```

```
    if i >= 55000 and i < 63000:
        datosIntervalo1 += 1
    elif i >= 63000 and i < 71000:
        datosIntervalo2 += 1
    elif i >= 71000 and i < 79000:
        datosIntervalo3 += 1
    elif i >= 79000 and i < 87000:
        datosIntervalo4 += 1
    elif i >= 87000 and i < 95000:
        datosIntervalo5 += 1
    elif i >= 95000 and i < 103000:
        datosIntervalo6 += 1
    elif i >= 103000 and i < 111000:
        datosIntervalo7 += 1
    elif i >= 111000 and i < 119000:
        datosIntervalo8 += 1
    else:
        datosIntervalo9 += 1

frecuenciasAbsolutasIntervalos.append(datosIntervalo1)
frecuenciasAbsolutasIntervalos.append(datosIntervalo2)
frecuenciasAbsolutasIntervalos.append(datosIntervalo3)
frecuenciasAbsolutasIntervalos.append(datosIntervalo4)
frecuenciasAbsolutasIntervalos.append(datosIntervalo5)
frecuenciasAbsolutasIntervalos.append(datosIntervalo6)
frecuenciasAbsolutasIntervalos.append(datosIntervalo7)
frecuenciasAbsolutasIntervalos.append(datosIntervalo8)
frecuenciasAbsolutasIntervalos.append(datosIntervalo9)
```

```
frecuenciasAbsAcumIntevalos = []
auxiliar = 0
for i in frecuenciasAbsolutasIntervalos:
    frecuenciasAbsAcumIntevalos.append(i + auxiliar)
    auxiliar= auxiliar + i

frecuenciasRelativasIntervalos = []
frecuenciasRelAcumIntervalos = []

for i in frecuenciasAbsolutasIntervalos:
    frecuenciasRelativasIntervalos.append(round((i / muestraTotal), 5))

for i in frecuenciasAbsAcumIntevalos:
    frecuenciasRelAcumIntervalos.append(round((i / muestraTotal), 2))

#Rellenamos las frecuencias de los valores y creamos una variable para las frecuencias.
listaFrecuenciasAbsolutas = [0]

#Creamos una variable iterador. Volvimos a valorizar el auxiliar con el primer dato de datosLista.
#Iteramos datosLista y para cada uno de sus elemento si son distintos al auxiliar, en la locación que tiene listaFrecuencia con índice(iterador) sumamos uno a la frecuencia.
#Sumamos uno al iterador una vez que no se repita más el número que tiene el auxiliar guardado.
iterador = 0
auxiliar = listaDatos[0]
for i in listaDatos:
```

```
    if i != auxiliar:
        auxiliar = i
        iterador += 1
        listaFrecuenciasAbsolutas.append(0)
    listaFrecuenciasAbsolutas[iterador] += 1

#Cálculo de frecuencias absoluta acumulada.
listaFrecuenciasAbsAcum = []
auxiliar = 0
for i in listaFrecuenciasAbsolutas:
    listaFrecuenciasAbsAcum.append(i + auxiliar)
    auxiliar += i

#Cálculos de frecuencias relativas y frecuencias relativas acumuladas.
listaFrecuenciasRelativas = []
listaFrecuenciasRelAcum = []

for i in listaFrecuenciasAbsolutas:
    listaFrecuenciasRelativas.append(round(i / muestraTotal, 5))

for i in listaFrecuenciasAbsAcum:
    listaFrecuenciasRelAcum.append(round(i / muestraTotal , 5))



intervalos = list(range(55000, 127000, 8000))

#Valores.
print("La muestra es de:",muestraTotal)
print("La sumatoria de frecuencias es:",sum(listaFrecuenciasAbsolutas))
```

```
print("La cantidad de elementos no repetidos son:", len(listaValores))

print("El valor mínimo es:", listaValores[0])

print("El valor máximo es:", listaValores[-1])

print("Datos agrupados en intervalos de 8000 km.")

print("Los autos que realizan la VTV con menos de 70 mil km son:",menoresA70)

print("El porcentaje de los autos que realizan la VTV con menos de 70 mil km
son:", porcentajeMenoresA70, "%")

print("Frecuencias por cada intervalo:",frecuenciasAbsolutasIntervalos)

print("Frecuencias acumuladas por intervalos:", frecuenciasAbsAcumIntevalos)

print("Frecuencias relativas por intervalos:", frecuenciasRelativasIntervalos)

print("Frecuencias relativas acumuladas por intervalos:",
frecuenciasRelAcumIntervalos)


#Cálculos de medidas de tendencias

print("Esta es la moda más pequeña:" , mode(listaDatos))

print("Esta es la mediana:" ,median(listaDatos))

print("Esta es la media:", round(mean(listaDatos),2))

print("Estas son todas las modas:", multimode(listaDatos), "C/U se repite 12
veces.")


#Cálculo de medidas de dispersión.

print("Esta es la varianza:" ,round(variance(listaDatos),4))

print("Este es el desvío estandar:", round(np.sqrt(variance(listaDatos)),4))

print("Este es el coeficiente de variación:", round((np.sqrt(variance(listaDatos))/
round(mean(listaDatos),2)) * 100, 2) , "%")

print("Este es el coeficiente de asimetría de pearson:", round(skew(listaDatos),4))

print("Posición de P sub 30 en la frecuencia absoluta acumulada.", (30 *
muestraTotal)/ 100)

print("P sub 30, se encuentra en el tercer intervalo.")

print("Por debajo de", round((71000 + ((((30 * muestraTotal) / 100) - 48157) /
23990) * 8000)), "se encuentra el 30%" , "de los datos.")
```

```
#Cierre del archivo.
miArchivo.close()



df = pd.DataFrame({"x": intervalos, "f": frecuenciasAbsolutasIntervalos, "F": frecuenciasAbsAcumIntevalos, "fr": frecuenciasRelativasIntervalos, "Fr": frecuenciasRelAcumIntervalos})
print(df)



intervalos=[55000, 63000, 71000, 79000, 86000, 95000, 103000, 111000, 119000, 127000]

plt.xticks(intervalos)
plt.hist(listaDatos, bins=intervalos)

y,edges,_=plt.hist(listaDatos, bins=intervalos, histtype='step', edgecolor='k')

midpoints=0.5*(edges[1:]+edges[:-1])
plt.plot(midpoints,y, 'r-*')
plt.show()
```